Responsiv webbdesign i LESS

för projektet ”Västankvarn – en västnyländsk matkälla”

Rasmus Sahlberg

Examensarbete för Tradenom -examen
Utbildningsprogrammet för Informationsbehandling
Raseborg 2014

EXAMENSARBETE

Författare: Rasmus Sahlberg
Utbildningsprogram och ort: Informationsbehandling, Raseborg
Handledare: Tomas Hammar
Titel: Responsiv webbdesign i LESS för projektet ”Västankvarn – en västnyländsk matkälla”

Datum 23.05.2014 Sidantal 48 Bilagor 0

Abstrakt

Det här arbetet beskriver utvecklingen av en webbsida med responsiv webbdesign åt Västankvarn Gårds projekt ”Västankvarn – en västnyländsk matkälla”. Projektets mål är att ta fram lokala och hållbara odlingslösningar inom odlings- och förädlingsverksamheten.Verksamhetstiden för projektet är tre år.

Webbsidans utseende har byggts upp från grunden, från skiss till färdig produkt, med CMS:et Joomla. För utseendekodning har använts kodspråket LESS, som möjliggör snabbare och mer flytande kodning samt redigering av redan skriven kod.

Planeringen och utförandet av webbsidans tema har gjorts med penna och papper, samt genom bildredigeringsprogrammet Adobe Photoshop. I det här examensarbetet diskuterar jag planeringen av temat och de två olika designerna, som gjordes åt beställaren.

Kraven var att skapa en webbsida med ett välkomnande utseende, samt att den skall fungera på både finska och svenska. Webbplatsen skall även ha en webbshop och en sociala medier gilla -länk samt ge allmänheten information om deras projekt. Kraven till webbsidan var några och på så sätt lätt att möta. Slutprodukten är bättre än förväntad och feedbacken har varit positiv.

Språk: Svenska
Nyckelord: Webbplats, Responsiv webbdesign, Västankvarn, Joomla, LESS

BACHELOR'S THESIS

Author: Rasmus Sahlberg
Degree Programme: Business Information Technology
Supervisor: Tomas Hammar
Title: Responsive web design with LESS for the "Västankvarn – en västnyländsk matkälla"-project / Responsiv webbdesign i LESS för projektet "Västankvarn - en västnyländsk matkälla"

Date 23 May 2014 Number of pages 48 Appendices 0

Abstract

This thesis describes the ongoing battle on how to design and produce a website with a responsive web design for Västankvarn Gård's project 'Västankvarn – en västnyländsk matkälla'. The purpose of the project is to do research in local and sustainable solutions in cultivation and breeding activities over the next three years.

The website is built from scratch, from an idea to a product the customer can use. For this thesis the CMS called Joomla has been used. The coding of the responsivity is done in a style sheet language called LESS, which enables faster and smoother coding as well as editing of already written code.

The planning and the creation of the website template is made by pen and with a graphics editing software called Adobe Photoshop. In this thesis I discuss the planning, of the template, and the two different web designs I made for my customer.

The requirements were to create an appealing website, which shall work in both Swedish and Finnish. The site shall also have a webshop and a way to like their social media pages as well as give public information about the project.The requirements were quite few and thus easy to meet. The outcome of this project is a success and well met by the customer and the feedback given is positive.

Language: Swedish
Keywords: Website, Responsive web design, Västankvarn, Joomla, LESS

Innehållsförteckning

Ordlista

CMS = Innehållshanteringssystem *(eng. Content Management System)*, används för att underlätta skapandet av webbsidor

CSS = *Cascading Style Sheets*, en typ av kodspråk för webbsidans utseende

DIV = område i HTML-koden för innehåll, står för engelskans *division*

GPL = *General Public License*, en licens för program som är baserade på öppen källkod

HTML = *HyperText Markup Language*, kodspråket för webbsidans struktur

LESS = *Leaner CSS*, en typ av kodspråk för webbsidans utseende

RWD = Anpassningsbar/responsiv webbdesign (*eng. responsive web design)*, som anpassar utseende åt sidan baserad på webbläsarfönstrets bredd.

SEO = Sökmotoroptimering *(eng. Search Engine Optimization)*. Sätt att se till att sökmotorer, som Google, hittar din webbsida med hjälp av nyckelord

Em = Breddenhet inom typografi. Em är alltid lika med webbläsarens fontstorlek i pixlar.

Back-end = Administrationspanelen på en webbplats

Front-end = Webbsidans utseende för allmänheten

Open source = Öppen källkod, gratis att använda och distribuera

Plugin = Instickspogram, installerbart program i CMS

Syntax = Regler, som definerar uppsättningen av reserverade ord och funktioner i ett programmeringsspråk

1 Inledning

Detta examensarbete handlar om utvecklingen av en hemsida med responsiv webbdesign åt Ulrika Grönvik, projektledare för projektet ”Västankvarn – en västnyländsk matkälla” på Västankvarn Gård i Ingå, Finland. Den engelska termen ”responsive web design”, även ofta förkortad som RWD, används också i det svenska språket. Förutom att hemsidan skall ha en responsiv webbdesign kommer den också att fungera på de två inhemska språken, finska och svenska. Webbsidan kommer också att inkludera en webbshop, som implementeras till hemsidan med hjälp av en färdig komponent. Komponenten modifieras sedan för att motsvara kriterierna.

Alla element på webbsidan kommer att fungera responsivt på olika upplösningar upp till en maximal bredd på 960 bildpunkter (pixlar/pixels). Därefter standardiseras bredden och ett bakgrundselement förblir fastklistrat i webbläsarens högra kant.

För utveckling av responsiv webbdesign användes LESS, som är nästlad/sammanfogad CSS (Cascading Style Sheets). Nästlad betyder att koden skrivs i sektioner istället för att skriva som i traditionell CSS. I traditionell CSS räknar man upp webbsideområdenas klass-värden och ger utseende åt dem skilt. I LESS försöker man undvika detta genom att använda kodspråkets färdigt inbyggda funktioner.

1.1 Syftet

Syftet med det här utvecklingsarbetet är att utveckla en användarvänlig webbsida för projektet. Webbsidan skall vara Västankvarn Gårds ansikte utåt för projektet, och marknadsföra projektet till utomstående. Målet med arbetet är självutveckling genom att använda de kunskaper och lärdomar, som har lärts ut i undervisningen i skolan och på fältet. Målet är också att använda moderna webbutvecklings- och webbdesignmetoder.

I det här arbetet diskuteras användningen av LESS för responsiv webbdesign, utvecklingen av responsivt tema genom ”media queries”, användningen av olika program och en kort introduktion till tre olika innehållshanteringssystem. Här ingår också en beskrivning på olika syntaxer, som det finns i LESS.

1.2 Västankvarn Gård

Västankvarn Gård finns i Ingå i Västra Nyland och ägs idag av Helsingfors universitet. Wilhelm och Lina Sandell donerade gården åt universitetet år 1894. Västankvarn Gård samarbetar idag med Yrkeshögskolan Novia och utbjuder en försöks- och undervisningsgård för naturbruksprogrammen vid yrkeshögskolan. Förutom samarbetet med Novia erbjuder gården också lantbruk med mjölkproduktion, skogsbruk och gårdsturism. Västankvarn Gård ordnar också olika evenemang såsom hundlägerverksamhet. Gården hyr också ut fastigheter.

1.2.1 Västankvarn – en västnyländsk matkälla

”Västankvarn – en västnyländsk matkälla” är ett projekt, som ordnas av Västankvarn Gård. Projektet varar i tre år och dess mål är att ta fram lokala och hållbara odlingslösningar inom odlings- och förädlingsverksamheten. På det här sättet hoppas de på att ge näringen en nyare, bättre och bredare grund att stå på.

Projektets andra syfte är att stöda och informera yrkesodlare, studeranden, nyutexaminerade och mångkulturella gruppper, samt att ta tillvara den kunskap som finns i Västnyland. Informationen gäller varornas ursprung, eftersom den kan spela en stor roll då man köper produkter. Projektets demonstrationsodlingar och vidareförädling ordnas på Västankvarn Gård.

1.3 Kravspecifikation

I början av projektet ordnades ett kundmöte och då bestämdes kraven för webbsidan. Till kraven hörde att webbsidan skall vara så användarvänlig som möjlig, se välkomnande ut, vara enkelt uppbyggd och ha en webbshop, som producenter kan sälja sina produkter genom. Även en användarmanual skall skrivas för att underlätta upprätthållningen av sidan.

Webbplatsen skall även vara lätt att upprätthålla, fungera på de två inhemska språken, finska och svenska, och det skall vara enkelt att lägga till och ta bort innehåll. Kundens första önskan var att både en webbshop och en hemsida skulle utvecklas. Dock insågs det snabbt att utvecklingen av en webbshop skulle ta för länge och utveckling av responsiv webbdesign erbjöds istället.

2 Innehållshanteringssystem (CMS)

I följande punkter diskuteras de tre mesta använda innehållshanteringssystemen på Internet; WordPress, Drupal och Joomla. WordPress och Drupal introduceras kort medan Joomla får en längre beskrivning, eftersom systemet är använt till det här projektet.

Ett innehållshanteringssystem *(eng. Content Management System)* är ett program som installeras på webbservern för att underlätta skapandet av webbsidor. Vid användning av ett CMS kan man även lätt skapa nya artiklar och andra element, på en webbplats. I och med att man använder ett sådant här webbsideprogram har även nybörjare en chans att utveckla webbsidor.

Det finns många fördelar med användningen av ett innehållshanteringssystem. Den största fördelen är att man lätt kan skapa nytt eller ta bort gammalt innehåll, detta är även till fördel för de som kan lite eller ingen alls HTML. Man kan också designa om webbsidans utseende utan att röra innehållet, eftersom utseendekoden är separat från innehållskoden.

Med hjälp av ett CMS har också webbplatsens innehavare bättre kontroll över sökmotoroptimeringen genom att ange sökord, specifika för webbsidan. Man kan även förenkla webbsidans inre adress, från den exakta adressen till en mer lättläst. Även om användningen av en sådan här programvara för med sig många fördelar, är den största nackdelen dock att programmet oftast kräver att de nyaste tekniska uppdateringarna finns installerade på webbservern. (Rob Tomlinson)

2.1 WordPress

WordPress är ett av de nyare CMS:en, som har publicerats och som används nuförtiden. Programmets utveckling började redan 2001, men systemet publicerades först år 2003. Idag är WordPress det världsledande CMS:et. WordPress är utvecklat i PHP och MySQL och licenserat under GNU GPLv2 eller senare. GPL står för *General Public Licens,* vilket betyder att programmet är licenserat under en upphovsrättslicens för gratis programvara och källkoden är på så sätt öppen. Innehållshanteringssystemet var från början byggt och avsett för bloggare, men idag gör det grund för mer än 68 miljoner webbsidor på Internet. (WordPress) Idag uppskattas 22,1 % av webbplatserna på Internet använda WordPress. (W3Techs)

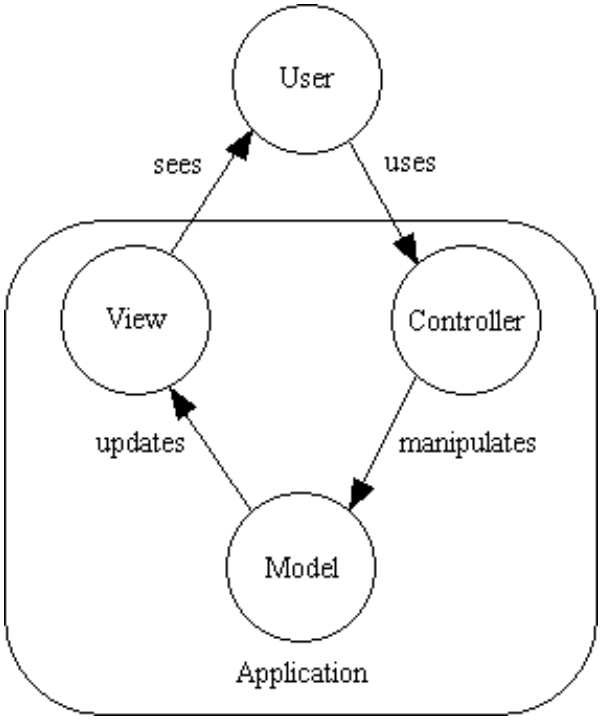
2.2 Drupal

Drupal publicerades i januari 2001 och är det äldsta innehållshanteringssystemet, som används än idag. Innehållshanteringssystemet är uppbyggt på PHP och MySQL. Drupal är liksom WordPress också distribuerat under GPL-licensering och används idag av mer än 630 000 människor. Systemet lämpar sig för privatpersoner, företag och organisationer. Till Drupals principer hör bl.a. utveckling, standarder och användarvänlighet. (Drupal) Idag uppskattas det att 1,9 % av alla webbplatser på Internet använder Drupal. (W3Techs)

2.3 Joomla

Liksom WordPress och Drupal är Joomla också ett gratis innehållshanteringssystem, som är utvecklat för konstruktion samt publicering av webbinnehåll och webbsidor. Systemet publicerades i augusti 2005 och har sedan dess växt till ett CMS med över 9400 tillägg och har översatts till över 60 olika språk. Joomla har laddats ner mer än 50 miljoner gånger och idag används systemet på 3,1 % av webbplatserna på Internet. Systemet är byggt enligt ”model-view-controller” (MVC) webbapplikation ”framework”-modellen, som också kan användas självständigt. (Joomla, W3Techs)

MVC betyder att ett program, med ett användargränssnitt, är installerat på en domän. Model står för domän eller data, vilket betyder att ett sådant måste finnas tillgängligt. View står för programmets användargränssnitt och Controller för inputlogiken, dvs. att man kan lägga till eller ta bort något med hjälp av programmet. I figur 1 förklaras det hur model-view-controller-modellen fungerar.



Figur 1. Model-View-Controller-modellen

Systemet är skrivet i PHP och använder sig av objektorienterad programmeringsteknik *(OOP = eng. Object Oriented Programming)* från version 1.5 och baserar sig på mjukvarudesign-mönster *(eng. software design pattern)*. Systemet kan lagra data i databaserna MYSQL, MSSQL från version 2.5 eller PostgreSQL från version 3.0. (Joomla)

2.3.1 Tekniska aspekter

För att den senaste versionen av Joomla skall kunna köras på en webbserver krävs minst följande tekniska behov. (Joomla)

- PHP 5.3.1 +
- MySQL (med stöd för InnoDB) 5.1 + eller
- MSSQL 10.50.1600.1 + eller
- PostgreSQL 8.3.18 +
- Apache 2.x +
- Nginx 1.0
- Microsoft IIS 7

Komponenter, insticksprogram *(eng. plugin)* och moduler kan ha avvikande tekniska krav för att de skall fungera. För att vara på den säkra sidan lönar det sig dock att kontrollera att

webbservern har stöd för de rekommenderade tekniska aspekterna, som CMS:et Joomla kräver.

2.4 Varför valdes Joomla för detta projekt?

Jag valde Joomla eftersom det var bekant från tidigare. Jag hade använt systemet i ett tidigare projekt i undervisningen. Den sex veckor långa praktiktiden på Sydweb, i Ekenäs under våren 2012, gav djupare inblick i innehållshanteringssystemet Joomla. Jag har också använt innehållshanteringssystemet i senare utvecklingsprojekt, som gjorts efter praktiktiden på Sydweb.

Den administrativa panelen i Joomla är, jämfört med Drupal, mycket användarvänligare. Utvecklingen av ett tema är också mycket enkelt och webbplatsens ”front-end”, dvs. utseendet utåt, är lätt att strukturera. Man kan även enkelt ge olika rättigheter och bestämma vad som skall visas på vilken sida. Visning av element och artiklar kan också begränsas till specifik användarkategori.

Joomla valdes på grund av dålig erfarenhet av Drupal och ingen som helst erfarenhet av WordPress. Om WordPress hade valts hade det betytt inlärning av ett nytt CMS och det skulle ha varit tidskrävande. Anledningen till att Drupal genast valdes bort är att systemet anses mycket klumpigt och användarvänligheten av administreringsverktyget är mycket låg. Både Drupal och WordPress var därmed enkla att utesluta.

3 Programmeringsspråk

Inom webbprogrammering, -design och –utveckling finns det många olika kodspråk, som man kan använda till olika syften. För det här arbetet valdes användningen av LESS, PHP, CSS och HTML. I följande punkter förklaras i korthet vad dessa språk är, deras funktioner samt för- och nackdelar.

3.1 LESS

LESS (Leaner CSS) är ett öppen källkods -baserat dynamiskt stilmallsspråk *(eng. style sheet language)* designat av Alexis Sellier, från Berlin, Tyskland. Språket är sedan utvecklat av honom själv och Dmitry Fadeyev. Kodspråket är påverkat av SASS

(Syntatically Awesome Style Sheets, även kallat Sassy CSS), vilket man ser eftersom språket liknar strukturen som CSS-språket använder. LESS-språkets första version var skriven i Ruby, men Alexis Sellier övergick till att använda JavaScript istället, eftersom användningen av Ruby ogillades av LESS-användarna (Wikipedia, lesscss). LESS är egentligen ett alternativt sätt att skriva CSS på.

Eftersom LESS är skrivet i JavaScript behöver man en less.js -fil för att kunna använda språkets syntaxer och funktioner. Dessa gör kodningen märkbart snabbare.

3.1.1 Kort introduktion till funktioner

Till skillnad från traditionell CSS ger LESS användaren möjlighet att implementera olika syntaxer och funktioner. Detta gör det lättare att, i senare skede, använda samma designstil, samt att t.ex. ändra på färgkoder och dimensioner, och samtidigt behålla förhållandet mellan både yttre och inre marginalerna på webbsidan.

Även om LESS till största delen skrivs som traditionell CSS har språket många fördelar. Användaren kan definiera en variabel i början av .less-dokumentet och sedan använda samma variabel genom hela dokumentet. Variabeldefinitionen görs med hjälp av ett @-tecken, som följs av en textsträng som användaren själv väljer. @-variablerna är LESS-kod och dessa kompileras över till traditionell CSS-kod, som illustreras i kodexempel 1.

Kodexempel 1. Traditionell CSS-kod

```
.content {color: #152333;}
.navigation {color: #AEBCCC;}
```

LESS har även den fördelen att man får ett felmeddelande om man har lämnat bort ett tecken eller har satt till ett ogiltigt.

3.1.2 Syntaxer i LESS

Det finns många olika syntaxer i LESS, och de mest användbara är; variabler, mixins, parameter-mixins, nästlade regler, operatörer, färgfunktioner, namnrymder, omfattningar och JavaScript-utvärdering. Dessa syntaxer finns också tillgängliga i SASS. I den här punkten beskrivs dessa syntaxer i den ordningsföljd som de nämndes. (SmashingMagazine)

En mixin är en klass åt en klass, medan en parametrisk mixin är en klass, som man kan ge en parameter åt. Med hjälp av nästlade regler kan man skriva klasser inom klasser, vilket hjälper till att man inte behöver repetera samma kod om och om igen. Syntaxen ”operator” är egentligen matematik inom CSS. Med färgfunktioner kan man lätt redigera färgerna i dokumentet. En namnrymd är en grupp av stilar, som man kan använda genom att hänvisa till dem. Omfattningen möjliggör lokala ändringar till stilar och JavaScript-utvärderingen möjliggör användning av JavaScript inom CSS. (SmashingMagazine)

En variabel i LESS utgörs av @-tecknet och följs av en textsträng, som kodaren själv definierar. Med hjälp av en variabel kan man t.ex. definiera olika färger. I kodexempel 2 definieras en färg med hjälp av en variabel, som sedan används igenom koddokumentet. Först anges textsträngen till variabeln och därefter definieras färgen till den. Färgen utgörs av en hexadecimal-kod. En hexadecimal-kod är en färgkod, som utgörs av #-tecknet följt av sex tecken från 000000-FFFFFF.

Kodexempel 2. Färganvändning i LESS med hjälp av en variabel

```
@text-color: #000000;
.content {color: @text-color;}
.menu {color: @text-color;}
```

Användningen av en mixin gör stilgivningen för flera olika klasser mycket smidig. Då man använder sig av mixin-syntaxen skall man först definiera ett utseende för en klass. Därefter kan man använda sig av den här klassen inom andra klasser, som beskrivs i kodexempel 3. Efter att koden har kompilerats till traditionell CSS, kommer områdena med klasserna ”.content” och ”.slideshow” ha en svartfärgad en-pixel-hög ram.

Kodexempel 3. Exempel på mixin-användning i LESS

```
.ux-menu {border: 1px solid #000;}
.content {.ux-menu}
.slideshow {.ux-menu}
```

En parameter-mixin är lämplig då man t.ex. anger rundade kanter till ett element. En parameter-mixin utgörs i stort sätt av en variabel. Tillämpar man en parameter-mixin med en vanlig mixin gör man kodningen av stilgivning mycket enkel. I kodexempel 4 beskrivs hur man använder en parameter-mixin, medan kodexempel 5 ger exempel på hur en parameter-mixin används inom en vanlig mixin. I kodexemplen 4 och 5 kommer hörnen att ha en radie på 10 pixlar.

Kodexempel 4. Användning av en parameter-mixin

```
.border-radius (@radius: 10px;) {
        -webkit-border-radius: @radius;
        -moz-border-radius: @radius;
        border-radius: @radius;
}
```

Kodexempel 5. Exempel på parameter-mixin i en vanlig mixin baserad på koden ovan

```
.content {
        .border-radius;
}
```

Till skillnad från CSS används nästlade regler, som selektorer i LESS. En selektor *(eng. selector)* används för att välja och måla underelement, för att kunna ge en stil åt dem. Då man vill ge en stil åt ett element, skrivs en selektor i CSS så att man räknar upp huvud- och underklasser efter varandra. I LESS görs detta lättare då man skriver en likadan selektor i nivåer i stället. I ett CSS-dokument är detta mycket svårläst och det är nästan omöjligt att hitta specifik kod. I kodexempel 6 illustreras hur selektorer skrivs enligt nästlad regel i LESS. I CSS skrivs samma selektor som ”.element1 .element2 .element3 .element4 h2”. Enligt kodexemplet har man även möjlighet att ge skilda stilar och utseenden åt varje element medan man arbetar sig mot h2-elementet i koden.

Kodexempel 6. Selektorer i LESS enligt nästlad regel

```
.element1 {
        .element2 {
                .element3 {
                        .element4 {
                                h2 { /*stil för h2 kommer här*/ }
                        }
                }
        }
}
```

En annan välanvändbar selektor i LESS är &-tecknet, som används för att välja elementens pseudo-element. Ett pseudo-element kan t.ex. vara attributet, som uppkommer då man för muspekaren över en klickbar länk. I det här fallet betyder &-tecknet detsamma som ”this” i JavaScript. (SmashingMagazine) I kodexempel 7 illustreras hur man använder &-tecknet för att välja en länks ”hover”-pseudo-element. Länken utgörs av bokstaven ”a” i kodexemplet.

Kodexempel 7. Användning av selektorn ”&”

```
a {...
        &:hover {....}
}
```

Som redan nämndes kan man använda matematik i LESS, i form av operatörer. Till de matematiska funktionerna hör de matematiska tecknen, såsom addition, subtraktion, multiplikation, division och parenteser. Den matematiska syntaxen i kodspråket följer också räknereglerna där en parentes alltid utförs först. (SmashingMagazine) I kodexempel 8 beskrivs hur man kan använda en matematisk funktion för att dela bredden på ett element i två. I samma kodexempel illustreras också användning av parentes i LESS. I kodexemplet definieras variabeln ”@full-page” först och sedan definieras variablerna ”@halfpage” och ”@quarter-page” genom att dividera ”@full-page” med talet två.

Kodexempel 8. Matematiska funktioner i LESS

```
@full-page: 960px;
@halfpage: @full-page / 2;
@quarter-page: (@full-page / 2) / 2;
```

Färgfunktionen i LESS är mycket mångsidig. Till syntaxen kan användas olika former av färgfunktioner. Dessa funktioner kan vara ”saturate”, ”desaturate”, ”lighten”, ”darken”, ”fadein”, ”fadeout”, ”fade”, ”spin”, ”mix”, ”greyscale” och ”contrast”. I tabell 1 förklaras det närmare vad dessa funktioner gör. Parametrar, som tas till dessa funktioner, är ”color” och ”amount”. ”Color”-parametern utgörs av ett hexadecimal-attribut, medan ”amount” definieras i procent. HSL står för ”Hue”, ”Saturation” och ”Lightness”. (LESS)

Tabell 1. Färgoperatörsfunktioner i LESS (LESS)

Saturate	Ökar på färgmättnaden av en färg i HSL-färgskalan
Desaturate	Minskar på färgmättnaden i HSL-färgskalan
Lighten	Gör en färg ljusare enligt HSL-skalan
Darken	Gör en färg mörkare enligt HSL-skalan
Fadein	Ökar på genomskinligheten
Fadeout	Minskar på genomskinligheten
Fade	Definierar det absoluta genomskinlighetsvärdet på en färg
Spin	Roterar ”hue”-vinkeln på en färg i en riktning
Mix	Smälter två färger samman
Greyscale	Tar bort färgmättnaden från en färg
Contrast	Väljer de två färger som ger bästra kontrasten mellan varandra

I kodexempel 9 illustreras användningen av ”lighten”-funktionen i LESS. ”Lighten”-funktionen tillämpas här på en textfärg.

Kodexempel 9. Illustrering av funktionen ”Lighten”

```
@text-color: #152333;
@menu-color: lighten(@text-color, 60%);
.navigation {color: @menu-color;}
```

I LESS finns det även funktioner för att sammansmälta färger. Dessa funktioner kan jämföras med dem som finns i bildredigeringsprogram, såsom Adobe Photoshop. Till dessa funktioner hör ”multiply”, ”screen”, ”overlay”, ”softlight”, ”hardlight”, ”difference”, ”exlusion”, “average” och “negation”. Dessa funktioner tar två olika färgparametrar, “färg1” och ”färg2”. (LESS) I tabell 2 beskrivs dessa funktioner och deras effekter närmare.

Tabell 2. Färgsammansmältningsfunktioner i LESS

Multiply	Multiplicerar två färgers rgb-färgkanaler, som sedan divideras med 255. Detta resulterar i en mörkare färg.
Screen	Motsattsen till ”multiply”. Resulterar i en ljusare färg
Overlay	Kombinerar effekten från ”multiply” och ”screen”
Softlight	Kan jämföras med ”overlay”
Hardlight	Samma som ”overlay”, men färgrollerna är de motsatta
Difference	Subtraherar färgen i den andra parametern. Negativa värden inverteras.
Exclusion	En liknande effekt som ”difference” men med en lägre kontrast
Average	Kalkylerar medelvärdet av två färger
Negation	Motsatten till ”difference”

Användningen av syntaxen ”namnrymd” i LESS möjliggör att man ytterligare kan lägga till en strukturnivå i koden. I kodexemplen 10 och 11 illustreras användningen av namnrymd-syntaxen i LESS. (SmashingMagazine)

Kodexempel 10. Definition av namnrymd

```
#namnrymd1 {
        .funktion () {
                color: #222;
                margin: 0; padding: .5em;
        }
}
```

När namnrymden har definierats kan man hänvisa till den då man vill använda den i ett senare skede av kodningen.

Kodexempel 11. Användning av namnrymd

```
.div1 ul {
        #namnrymd1 > .funktion;
}
```

Omfattningssyntaxen möjliggör användning av samma variabeltextsträng, men olika värden på olika ställen i kodfilen. Man kan alltså definiera samma variabeltextsträng två eller flera gånger, men också använda dessa skilt. (SmashingMagazine) I kodexempel 12

beskrivs hur detta fungerar. Kommentarer till hur variabeldefinitionen fungerar utgörs av /* och */.

Kodexempel 12. Definition av samma variabel

```
@color: #000; /*svart färgkod*/

.menu {
        @color: #fff; /*vit färgkod*/
        background: @color; /*bakgrunden är vit*/
}

.footer {
        background: @color; /*bakgrunden är svart*/
}
```

3.1.3 Snabbare kodning

I underkapitlet 3.1.2 beskrevs syntaxerna i LESS och hur man använder dem. Användningen och utnyttjandet av dessa till deras fulla potential möjliggör snabbare kodning och justering av koder blir också lättare i framtiden. Kodandet blir också mer organiserat och det krävs inte lika stor ansträngning till att hitta eller ändra på kod.

3.1.4 Användning av ”Media Queries” i LESS

I LESS kan man använda CSS3-funktionen ”media queries” på två olika sätt. Man kan skriva den här funktionen som i traditionell CSS eller enligt nästlade regler i LESS. I kodexempel 13 definieras först bredderna, som utseendet skall anpassas till. Följande tre kodexempel bygger på varandra.

Kodexempel 13. Definition av bredd med hjälp av variabler i LESS (stackoverflow)

```
@mobile: ~”screen and (max-width: 320px)”;
@desktop: ~”screen and (min-width: 980px)”;
```

I kodexempel 14 beskrivs hur man skriver en ”media query” –funktion enligt traditionell CSS-struktur. I det här kodexemplet ges en stil åt klasserna ”logo” och ”footer”.

Kodexempel 14. Media queries enligt traditionell CSS-struktur

```
@media @mobile {
        .logo {color: #ffffff;}
        .footer {width: 100%;}
}

@media @desktop {
        .logo {color: #000000;}
        .footer {width: 960px;}
}
```

Detta kan snyggas upp en del genom att skriva det enligt LESS-kodens nästlade regel. I kodexempel 15 illusteras detta genom att endast ge en stil åt ”footer”-klassen. Samma idé gäller för alla elementklasser och –id:n. I det här kodexemplet placeras klassen först och funktionen ”media queries” bakas in i klassen. Stilen ges sedan inom ”media query” –taggarna.

Kodstruktureringen i kodexempel 15 gör att de olika skärmupplösningsstilarna hålls på ett ställe. Den här typen av strukturering möjliggör också att man kan ge olika stilar, som tillämpas för alla utseenden i klasstaggen. Dessutom kan man också specificera avvikande utseende inom ”media query” –taggen. (Stackoverflow)

Kodexempel 15. Media queries enligt nästlad regel

```
.footer {...
        @media @mobile {
                width: 100%;
        }
        @media @desktop {
                width: 960px;
        }
}
```

3.2 CSS

CSS står för Cascading Style Sheets och är utvecklat av Håkon Wium Lie, Bert Bos och W3C (World Wide Web Consortium). Språket publicerades i december 1996 (Wikipedia). Istället för att använda sig av primitiv HTML-kod använder webbdesignern istället CSS, som möjligger definition av hur varje element på webbsidan visas. Genom att använda sig

av CSS kan man utseendemässigt göra mer komplexa webbutseenden än med ren HTML. (Wyke-Smith, 2006, s6)

Fördelarna med CSS är att det sparar tid, webbsidan laddas fortare, det är lätt att underhålla den och man har tillgång till fler möjligheter än i HTML. Den största nackdelen är dock webbläsarkompatibiliteten. Webbläsarkompatibilitet har alltid varit ett stort diskussionsområde inom webbdesign, eftersom Internet Explorer har varit efter i utvecklingen.

3.3 SASS

SASS står för Syntetically Awesome Style Sheets och möjliggör många fler funktioner än LESS gör. Språket publicerades år 2007 och är strukturerat av Hampton Catlin och utvecklat av Nathan Weizenbaum och Chris Eppstein. SASS är påverkat av CSS, YAML (YAML Ain't Markup Language) och Haml (HTML Abstraction Markup Language). Kodspråk som har påverkats av SASS är LESS, Stylus och Tritium. (Wikipedia) I kapitlet 3.5 beskrivs skillnaderna och likheterna mellan LESS och SASS.

3.4 Skillnader mellan LESS och CSS

Den största skillnaden mellan traditionell CSS och LESS är användningen av variabler. Variabelanvändningen gör kodningen lättare och justering av en textfärg kan göras med att endast ändra färgkoden på en plats i stället för på flera. I ett större CSS-dokument har man ofta samma hexadecimal (hex) -färgkod på flera olika ställen, vilket försvårar ändringen av dessa. Ju fler hexadecimal-färgkoder man har desto mer tidskonsumerande och krävande blir det. Tiden, som man använder till att ändra existerande kod, skulle man i stället kunna lägga ner på annan utveckling eller kodning.

En enkel ändring, som textfärgen, är inte den enda fördelen som man har användning av i LESS. I en webbsida är förhållandet mellan marginal och text mycket viktigt och man kan enkelt justera detta med hjälp av LESS, genom att använda sig av addition-, subtraktion-, multiplikation- och divisionstecken. Samma princip gäller när man bestämmer förhållandet mellan olika färger, textstorlekar osv.

3.5 Skillnader och likheter mellan LESS och SASS

SASS och LESS erbjuder i stort sätt samma funktioner, men SASS är vidareutvecklat och erbjuder på så sätt på många fler användningsmöjligheter. Förutom de syntaxer, som redan har nämnts för LESS erbjuder SASS också villkorssatser och kontroller. En villkorssats utgörs av ”if {} else {}”, ”for” och stöder villkorssatsoperatörerna ”and”, ”or”, ”not”, ”<” (mindre än), ”>” (större än), ”<=” (mindre än eller lika med), ”>=” (större än eller lika med) och ”==” (lika med). Matematiksyntaxen i SASS är också vänligare och mer utvecklad än den är i LESS. SASS har också fler färgfunktioner än LESS. Pseudo-element-funktioner i SASS utgörs av ”$” medan den i LESS utgörs av ”&”. (SmashingMagazine)

I kodexempel 16 beskrivs användning av villkorssatser i SASS. Här kontrolleras om ljusstyrkan är större än 30 % och om påståendet stämmer blir bakgrunden svart. Stämmer påståendet inte blir bakgrundsfärgen vit istället.

Kodexempel 16. Villkorssatser i SASS (SmashingMagazine)

```
@if lightness($color) >; 30% {
  background-color: #000;
} @else {
  background-color: #fff;
}
```

Som redan nämndes är de matematiska funktionerna i SASS vänligare eftersom man kan använda olika måttenheter. SASS kan räkna ihop de olika måttenheterna medan LESS inte klarar av det. I kodexempel 17 beskrivs detta.

Kodexempel 17. Användning av måttenheter i SASS och LESS (SmashingMagazine)

```
/* Sass */
2in + 3cm + 2pc = 3.514in
/* LESS */
2in + 3cm + 2pc = Error
```

3.6 PHP

PHP stod ursprungligen för Personal Home Page, personlig hemsida, och är utvecklat av Rasmus Lerdorf år 1994. Idag står PHP dock för Hypertext Processor och är ett server skriptspråk. Hypertext Processor betyder att PHP förbehandlar skriven data innan den blir

HTML. Fördelarna med PHP är att det är snabbt samt smidigare än ren HTML. Man kan konstruera funktioner för att imponera på webbsidebesökarna. Till skillnad från CGI (Common Gateway Interface), ASP (Active Server Pages) och JSP (JavaServer Pages) sparar programmeraren tid om man använder sig av PHP istället. (Ullman, 2002, s. x, xii)

3.6.1 JDocumentHTML

PHP-språket utvecklades för konstruktion av dynamiska webbsidor, vilket det används mest till nuförtiden. Joomla-projektteamet har utvecklat några egna dokumentkoder, som används som PHP. Dessa koder kan endast användas i CMS:et Joomla. Den här typen av kod kallas JDocumentHTML, varav den mest användbara, för webbsidor med basfunktioner, är countModules. Anledningen till att den här typen av kod tas upp under PHP-funktioner är att JDocumentHTML används inom PHP-taggar.

I kodexempel 18 kontrolleras det om modulen ”breadcrumb” finns och är aktiverad. Den här kontrollen sker på varje sida, som finns på webbplatsen. Om modulen är aktiverad skriver PHP-koden ut HTML-koden, som finns inom PHP-taggarna. Är modulen inte aktiverad för att visas på en viss sida skrivs den inte ut. Meningen med den här koden är att webbutvecklaren kan själv bestämma om en modul skall visas på en viss sida.

Kodexempel 18. Användning av countModules

```
<?php if($this->countModules('breadcrumb')): ?>…htmlkod…<?php endif; ?>
```

3.7 HTML 4.1

HTML står för HyperText Markup Language och är grunden för strukturen på en webbsida. Språket är utvecklat av Sir Timothy John ”Tim” Berners-Lee, en brittisk datavetenskapsman, på 1990-talet. (Infomesh)

Fördelarna med HTML är att det är enkelt att använda och lätt att lära sig. Språket är dagens webbstrukturstandard och stöds av alla webbläsare. Nackdelarna är att det är statiskt och man kan inte producera dynamiska webbsidor med endast HTML-kod, utan måste använda andra kodspråk. I projektet har använts HTML4.1, men den nyaste versionen är HTML5. Den nyare versionen möjliggör fler kodalternativ och tekniska lösningar.

4 Använda program

I följande punkter berättas det kortfattat om de program och funktioner, som har använts till detta projekt. Deras historia samt för- och nackdelar tas också upp.

4.1 Adobe Photoshop CS6

Adobe Photoshop är ett bildredigeringsprogram och den första versionen lanserades 1988 av Thomas Knoll (Wikipedia). Programmet har sedan dess utvecklats till ett avancerat program som används av grafiker idag. Det är dock inte gjort för nybörjare eftersom det blir och är en aning avancerat ju djupare man går in i dess funktioner och desto mera man lär sig. I det här projektet användes Adobe Photoshop CS6. CS står för Creative Suite.

4.1.1 För- och nackdelar

Jag använder själv Photoshop för att det är det bästa bildredigeringsprogrammet, som jag har stött på och lärt mig använda. Programmet är bra för att man kan uttrycka sin kreativitet på många olika sätt, utveckla olika designprojekt såsom webbsideteman.

Man kan även återställa gamla fotografier, kombinera grafik med text, göra konst med målpenslar, ändra färgen på fotografier, korrigera misstag i ett fotografi, designa kläder, ändra ett fotografis intryck konstnärligt och samtidigt känna sig väl till mods för att man har lyckats göra något artistiskt. (Marvi Ocampo)

Programmet har också den fördelen att man alltid lär sig något nytt. Dock är en av de största nackdelarna priset och programmet är på så sätt slutet, dvs. källkoden är inte öppen för alla utan endast programmets utvecklare.

4.2 Notepad ++

Notepad++ är ett öppen källkods -baserat textredigeringsprogram, som kollar kodspråkets syntax och färgar koden på basen av syntaxen. Notepad++ är skrivet i C++ och utvecklat av Don Ho (Notepad++). Första versionen av programmet publicerades i november 2003 (Wikipedia). Fördelarna med programmet är att det är gratis, lättbyggt och lätt att använda. Man måste även skriva ut all kod själv. Detta gör det lättare att bli en bättre kodare,

eftersom man lättare kommer ihåg den kod man själv har skrivit. Nackdelarna med programmet är att det inte ger kodalternativ eller fyller i resten av koden.

4.3 WAMP-Server

WAMP är ett öppen källkods -baserat program och står för Windows Apache MySQL PHP (Python eller/och PERL). Programmet möjligger en lokal webbservermiljö i operativsystemet Windows. WAMP ger användaren tillgång till de fyra viktigaste hörnstenarna i en webbserver; operativsystemet, databaser, webbservern och mjukvaror för webbskript. (Webopedia)

4.4 Andra program

För utvecklingen av webbsidan har även andra program använts för att utföra en del uppgifter. I följande punkter berättas det kort om dessa program och till vilket syfte de har använts till.

4.4.1 FileZilla

FileZilla är ett gratis FTP (File Transfer Protocol) -program, som används till att flytta filer från sin egen dator till en webbserver. Det finns två installerbara filer, en klient och en server, som man kan ladda ner från FileZillas egen hemsida. Skillnaden mellan dessa är att man använder klienten till att flytta filer och servern om man vill göra filer synliga för andra internetanvändare (FileZilla). Detta program användes till att flytta över filer till webbhotellets webbserver i slutet av projektet.

4.4.2 PHPMyAdmin

Tobias Ratschiller började år 1998 skapa PHPMyAdmin, som är ett öppen källkods -baserat verktyg, skrivet i PHP. Detta verktyg används för att lagra, skapa och ta bort databaser och/eller databasfiler. Programmet är designat för att administrera MySQL-databaserna på Internet. Man kan även skapa och ta bort bl.a. tabeller, kolumner, användare och rättigheter från databasfiler. Programmet har översatts till 72 olika språk. (phpMyAdmin) PHPMyAdmin användes för att lagra CMS:ets databasfiler.

4.4.3 Google Chrome

Google Chrome är företaget Googles egen webbläsare och är huvudsakligen den som har använts till det här projektet. För att skapa webbsidans utseende i LESS har webbläsarens färdigt inbygga inspektörverktyg använts. Det här verktyget ger möjligheten att skriva in CSS-kod för att se hur webbsidans olika element beter sig innan man sätter in koden i den egentliga kodfilen.

4.4.4 Mozilla Firefox

Mozilla Firefox är en webbläsare, som är utvecklat av företaget Mozilla. Denna webbläsare användes endast för att kontrollera att allt fungerar i den och att utseendet inte avviker från Google Chrome.

4.4.5 Internet Explorer

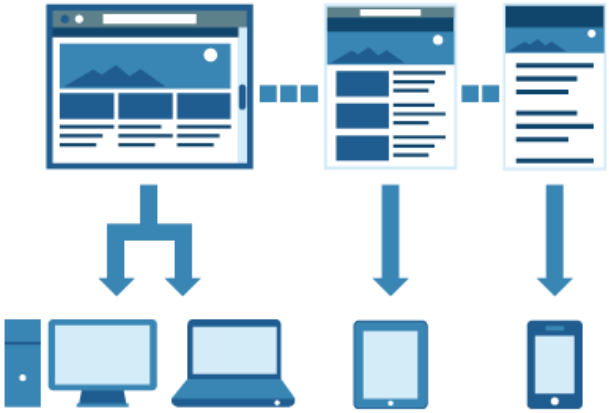
Internet Explorer, ofta förkortat som IE, är en webbläsare utvecklad av Microsoft. Denna webbläsare användes inte till annat än kod- och utseendeoptimering. Utseendet optimerades så att det även fungerar i version 7 och 8 av Internet Explorer. En mer detaljerad beskrivning hur optimeringen fullföljdes kan man läsa i punkt 8.3.

5 Responsiv webbdesign

Fenomenet responsiv webbdesign är den nyaste trenden inom webbdesignvärlden. Alla talar om det och de flesta webbsidor i dagens läge fungerar enligt den här tekniken. Men vad är det egentligen och hur framställer man en sådan här webbsida?

5.1 Vad är responsiv webbdesign?

I korthet är responsiv webbdesign en blandning av dynamisk och flytande *(eng. fluid)* design. Responsiv webbdesign är ett utseende, som lämpar sig för alla typer av terminaler. Den här typen av design använder sig av HTML-kodens div-kod istället för en tabellform, som redan har slopats av webbutvecklare för länge sedan. Div står för engelskans division, som betyder uppdelning. Div-koden gör det lättare att dela upp sidan i olika block och justera deras bredd och placering med hjälp av CSS-, LESS- eller SASS-kod. Figur 2 beskriver hur webbsidans utseende optimeras för olika terminaler.



Figur 2. Beskrivande bilder på responsiv webbdesign

5.2 Varför responsiv webbdesign?

Före responsiv webbdesign blev populärt inom webbdesign använde man sig av två olika sidor, en som var gjord endast för mobiltelefoner och en annan som var utvecklad för datorskärmar. Ur siduppehållarens synvinkel var det här systemet tidskrävande och inte alltid den bästa lösningen. I och med att man idag använder sig av responsiv webbdesign behöver man endast uppdatera innehållet på en version istället för två. Responsiv webbdesign är också SEO-vänligt. Man använder sig också av den här typen av design för att ge användare den bästa möjliga användarupplevelsen på alla typer av tekniska enheter hon eller han kan tänkas besöka sidan med. (Jared Chelf)

6 Västankvarnprojektet

Projektet att utveckla en hemsida med responsiv webbdesign åt grönsaksavdelningen på Västankvarn Gård har skett i sex olika stadier. Det första stadiet gick ut på att skissa upp ett utseende till webbsidan samt att hålla ett kundmöte för att få ut så mycket information som möjligt av beställaren. Ju mera information man har i början av ett projekt desto lättare är det att jobba och konstruera en produkt. Så snart det första stadiet var avklarat gick projektet över till andra stadiet.

Det andra stadiet gick ut på att leta efter olika slags bilder för att få ytterligare inspiration till webbsidans utseende. I det här skedet påbörjades även övergången från skiss till bild i Adobe Photoshop. Det tredje stadiet var relativt kort eftersom det endast gick ut på att dela

in utseendet, skapat i Photoshop, i sektioner för att sedan börja koda temakoden. Så snart temakoden och webbsidans olika områden var skapade påbörjades stilgivningen. Kodningen var fjärde stadiet i projektet.

Stadierna fem och sex utgjordes av uppladdning till webbservern samt felsökning och kodoptimering. Optimeringen gick ut på att få .less-dokumentet så litet som möjligt genom att dra ihop samma typ av kod, som upprepades flera gånger på olika platser.

6.1 Utveckling av tema

För att utveckla hemsidans utseende och struktur användes Adobe Photoshop och vid kodningen, dvs. övergången från bild till fungerande applikation, Notepad++. Då beställaren hade godkänt förslaget för utseendet började utvecklingen av det. Först delades utseendet in i sektioner för att underlätta kodningen. Därefter kodades basstrukturen upp enligt nutidens webbstandarder.

HTML-kodens .. -taggar användes till basstrukturen, tillsammans med JDocumentHTML samt PHP, och kodades i filen index.php. För att underlätta stilgivningen användes klass- subjektet för att ge namn åt diven. Detta namn användes sedan för att kalla på div-taggen i LESS-koden.

I index.php användes även PHP-kod för att öka dynamiken på webbsidan ”Västankvarn – en västnyländsk matkälla”. Joomlas färdigt inbyggda system användes för att kolla om en besökare är inloggad eller ej. Det färdigt inbyggda systemet möjliggör också att skilja på registrerade användare och dela in dessa i grupper. Det är sedan möjligt att ge de olika grupperna skilda rättigheter för att visa olika element på webbsidan. I kodexemplen 19 och 20 förklaras hur man med hjälp av PHP-kod kan man specificera elementvisning på basen av användarens eller användarklassens ID, som definieras i MySQL-databasen.

Kodexempel 19. Användning av användarklassens ID för visning av element

```
<?php
        $user =& JFactory::getUser();
        $user =& JFactory::getUser($user_id);
        $groups = isset($user->groups) ? $user->groups : array();
?>
```

Kodexempel 20. Förklaring hur kodexempel 16 kontrollerar en databas array

```
<?php
        if (in_array(9, $groups)){
         //echo 'only visible for guests';
         echo '<jdoc:include type="modules" name="navigation_guest" />'; }
 ?>
```

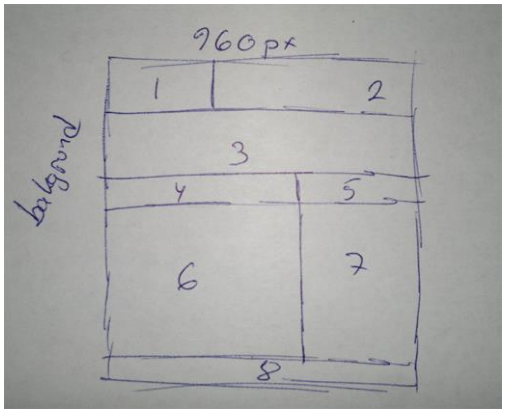
När basstrukturen var uppbyggd påbörjades kodningen av utseendet. Som redan nämndes användes kodspråket LESS. LESS-koden kompilerades sedan om till traditionell CSS med hjälp av ett systeminstickprogram, som heter ”LESS compiler”.

Basstrukturen kodades först så att sidans utseende såg grafiskt rätt ut och fungerade felfritt i alla webbläsare förutom Internet Explorer. Då den responsiva webbdesignen fungerade i Google Chrome, Mozilla Firefox och Opera, påbörjades optimeringen för Internet Explorer. Optimeringen för den här webbläsaren går ner till den sjunde versionen (IE7).

Som med alla webbsideprojekt kan det hända att kunden och designern har olika synsätt på saker och ting och den föreslagna designen kan på så sätt få göras om. Så gick det även i detta fall. I följande punkter diskuteras de två olika utseendena, i detalj, samt en jämförelse mellan dessa.

6.1.1 Skissen

Skissen är den första fasen i varje utveckling av webbplatser. Det är här man klottrar ner alla idéer man kan komma på och försöker sedan strukturera dem så att de blir en helhet. Dock finns det vissa element som nästan alltid är på samma ställe på varje webbplats på Internet, så även på mina egna sidor. Figur 3 beskriver den grafiska skissen och placeringen av logon (1) och språkval (2), om sådana finns. Till skissen hör även bildkarusell (3), meny (4), social medieikoner (5), innehåll (6), höger kolumn (7) och fotnot (8).

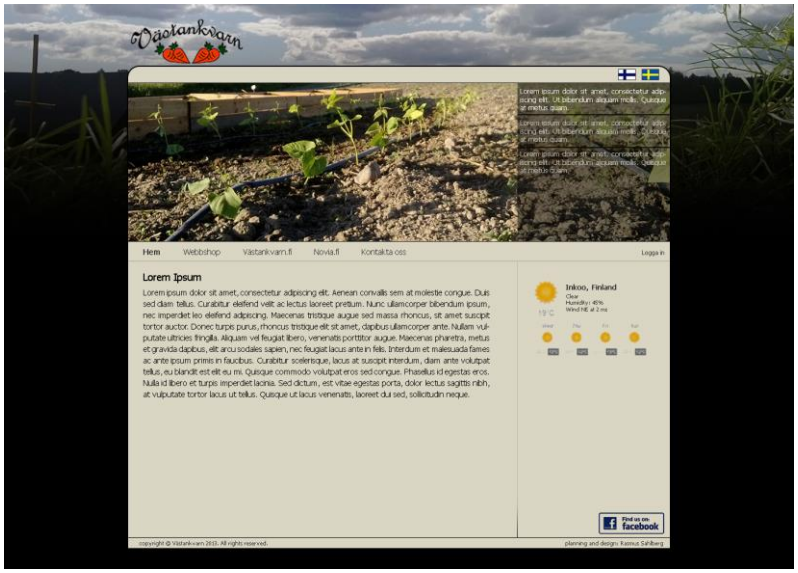


Figur 3. Tidig designskiss

Ett fåtal skisser gjordes upp och den enklaste av dem användes, för att få en bra struktur i helheten. Först skissades hela webbsidans utseende upp och sedan skapades skisserna för responsiv webbdesign, från mobiltelefon till datorskärm. Då skissen för responsiv webbdesign gjordes upp föreställdes hur en webbsida kan se ut i de olika bredderna och hur den sedan kommer att leva på basen av fönstrets bredd.

6.1.2 Första designen

Den första designen, figur 4, gjordes i början av projektet och hade en svart bakgrund, med en bakgrundsbild tagen från Västankvarn Gård. Bakgrunden på webbsidans innehållscontainer var en ljusgul/beige färg, färgkod: #dbd6c3. Färgen på texten var hel svart, färgkod #000000.



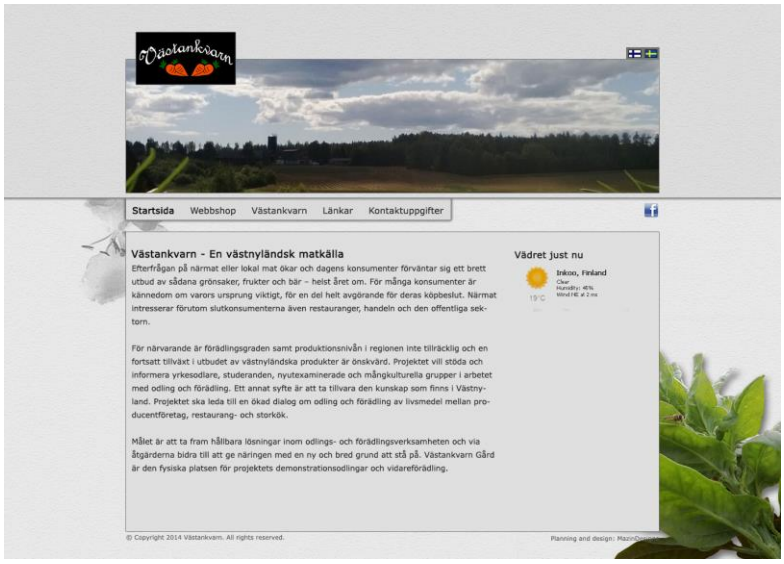
Figur 4. Första designen

Utseendeförslaget bestod också av en logo, språkvalsalternativ, bildkarusell, meny, ett innehållsområde och en höger kolumn. Till utseendet hörde också en fotnot. Tyvärr var den här designen för mörk och därför blev skapades det andra utseendet.

6.1.3 Andra designen

Det andra utseendet baserade sig på samma designskiss, som första utseendet, men är mycket ljusare än det första. Bakgrunden på den andra designen är konstruerad av två olika nyanser av en ljusgrå färg, en ljusare och en mörkare. Den mörkare nyansen är placerad i webbsidans övre del. Förutom att bakgrundsfärgen har två nyanser, så är den även skapad med funktionen ”Brevpapper” *(eng. Note Paper)* i Adobe Photoshop. Den här effekten gör att bakgrunden ser ut som ett skrovligt papper. Bakgrundens nyanser avgränsas med ett en-pixel-högt sträck, som sträcker sig över hela fönstrets bredd.

Till bakgrunden hör även två andra element, en blomma och en grönväxt. Blomman är placerad till vänster bakom menyn samt innehållselementet. Grönväxten är placerad i nedre högra hörnet med hjälp av ”position: absolute;”. Positionsvärdet ”absolute” frigör diven och den kan på så sätt flyttas fritt inom ett huvudområde. Samma positionsvärde har även använts för att placera blomman. Se figur 5 för den andra designen och elementens placeringar.



Figur 5. Andra designen

6.1.4 Designjämförelse

De två designerna är utseendemässigt helt olika, men ändå baserar sig båda på samma basstruktur. Båda utseendena har logo-elementet och språkvalsalternativen på samma plats. Bildkarusellen är litet smalare på höjden i den andra designen än den är i den första. I figur 6 jämförs utseendena med varandra. Här ser man även höjdskillnaderna och de andra grafiska skillnaderna mellan de två olika utseendena.

Figur 6. En sida-vid-sida jämförelse av utseendena

Textens typsnitt (font) är också ändrat från Tahoma till Verdana och hur fontens storlek renderas är även ändrad från pixlar till em. Detta betyder att storleken på texten alltid baserar sig på webbläsarens inställningar istället för att vara fixerad. Radavståndet på innehållstexten är 1.5 i respektive design.

Båda utseendena har en innehållscontainer med en bredd på 960 pixlar, och dess höjd utvidgas på basen av hur mycket text det är i den. Innehållets inre container är även kodad så att temakoden kollar om det finns något i den högra spalten. Om det inte finns något där förblir containern 960 pixlar bred, men om det finns innehåll i högra spalten förminskas bredden på innehållets container, till 650 pixlar, med en höger marginal på 20 pixlar, med hjälp av LESS. I kodexempel 21 förklaras hur detta kontrolleras.

Kodexempel 21. PHP-kod, som placeras i början av filen index.php

```
<?php
        $right = ($this->countModules('right'));
?>
```

I kodexempel 22 tillsätter PHP-koden ett ”narrow”-värde till CSS klass-attributet endast om PHP-påståendet ”if($right)” är sant, alltså lika med ett. Om påståendet är falskt används endast ”content”. Dessa två klass-värden används som skilda stilar i LESS-koden, ”.content” och ”.content.narrow”.

Kodexempel 22. Innehållskontroll för höger kolunm samt tillsättning av klass-värde

```
<div class="content<?php if($right) echo(' narrow') ;?>">
        <jdoc:include type="message" />
        <jdoc:include type="component" />
</div>
```

I LESS-filen kodades utseendet som nästlad kod. Fördelen med att skriva koden i nästlad form är en bättre struktur och organisering. Då man skriver nästlad LESS-kod kan pseudo-element-syntaxen ”&” användas. Syntaxen är ett sätt att repetera huvudklassen utan att skriva ut den. I traditionell CSS skrivs kodexempel 23 ut enligt två skilda stilar, ”.content” och ”.content. narrow”.

Kodexempel 23. Förklaring på nästlad LESS-kod

```
.content { width: 960px;
        &.narrow {
            width: 650px;
            margin: 0 20px 0 0;
        }
}
```

6.1.5 Index.php

I denna punkt beskrivs kodningen av webbsidans tema mer i detalj än vad som gjorts tidigare. Här diskuteras även de olika funktionerna, som har använts för att skapa temat till webbplatsens ”front-end”.

Webbsidans utseende består av tre olika huvudfiler; index.php, templateDetails.xml och en fil för utseendet. Den nödvändigaste är index.php, där strukturen kodas. I templateDetails.xml placeras modulernas positioner, som sedan väljs från och aktiveras i ”Module Manager” i Joomlas administrationspanel. Den tredje och sista filen är utseendet, som kan bestå av antingen en .css-, .less eller .sass-fil.

Index.php-filen består av div-taggar och är indelad i områden. De yttre ramarna för webbsidan utgörs av en div kallad ”wrapper”. Inne i den här diven finns det en div kallad ”pagesetup outer” för att vara säker på att vissa funktioner fungerar. I diven ”fullsite” placeras sedan webbsidans huvudsakliga element och de delas sedan upp i respektive områden för att göra LESS-kodningen så smidig som möjlig. Divarna får namnges enligt webbdesignerns eget tycke.

För att rendera olika funktioner på webbsidan användes så kallad ”jdoc include”-kod. Jdoc står för Joomla document. I den här koden skall det specificeras om det är frågan om en komponent eller modul, samt vilken stil man vill att området skall renderas som. Jdoc-koden placerades inom respektive divar, som sedan stilgavs. Webbsidans huvudsakliga innehåll renderas också av en jdoc-kodsnutt. I kodexempel 24 beskrivs hur en ”jdoc”-kod implementeras till webbsidans strukturkod.

Kodexempel 24. Implementering av Jdoc-kod

```
<div><jdoc:include type="modules" name="slideshow" style="xhtml" /></div>
```

För att kunna koda en klistrad fotnot placerades diven ”footer” utanför ”wrapper”-diven, och stilgavs sedan med hjälp av LESS-kod. Även diven med grönväxten placerades under ”footer”-diven för att sedan placeras i rätt position på webbsidan.

6.2 Moduler, komponenter och insticksprogram

För att göra webbsidan så dynamisk och användarvänlig som möjligt användes några färdigt utvecklade moduler, komponenter och insticksprogram. En modul kan i princip vara vad som helst på webbsidans ”front-end”, medan komponenter och insticksprogram installeras till CMS:et från webbplatsens administrationspanel (dock kan även moduler installeras). De följande punkterna beskriver dem som användes i det andra utseendet.

6.2.1 Logo

Webbsidans logo består av en bild, som är placerad i en, i CMS:et, färdigt inbyggd modul kallad ”Custom HTML”. Modulens jdoc-kod är placerad inne i en div, som är 205 pixlar bred och 87 pixlar hög. Den här modulen är sedan aktiverad och placerad uppe i vänstra hörnet på webbsidan. Diven är placerad så att nedre halvan av dess höjd ligger på

bildkarusellen och övre halvan utanför. Dessutom är diven indragen från vänstra sidan med 30 pixlar. Bilden i modulen är placerad i mitten, både horisontellt och vertikalt. Bakgrundsfärgen på logon är helsvart (färgkod #000000).

Webbsidans logo designades redan under våren 2013. Logon består av texten Västankvarn i en båge ovanför två par morötter, som är riktade mot varandra. Den här logon finns också i vektorformat och är även tryckt på grönsaksavdelningens arbetskläder.

Figur 7. Logons placering och utseende

6.2.2 Språkval

Ett av webbsidans många krav var att den skulle fungera på de två inhemska språken. För att åstadkomma detta användes gratisversionen av översättningskomponenten Falang, som är utvecklad av fransmannen Stéphane Bouey. Språkvalet utgörs av de finska och svenska flaggorna, som är placerade på en bakgrund med färgen #737373. Båda flaggorna är 20 pixlar breda och har en svart ram med en bredd på en pixel.

6.2.3 Bildkarusell

Bildkarusellmodulen, som används på webbsidan, heter ”Slideshow CK”. Modulen är enkel att använda och det är lätt att ladda upp egna bilder eller videon, som sedan visas med hjälp av den. Komponenten erbjuder även olika typer av effekter och alternativ för stilar. Dessutom kan man även ladda ner olika tillägg till Slideshow CK. Med dessa tillägg kan man ge tilläggsmöjligheter för hur komponenten laddar bilderna till front-enden.

Bildkarusellens storlek på webbsidans ”front-end” är 960 pixlar bred och 240 pixlar hög. Diven visas som ett block på sidan och har en en-pixel hög övre och undre ram, vars färg är #737373.

6.2.4 Vädret

På webbsidan kan man följa med väderleken i Ingå. Vädret hämtas av en modul som heter Weather GK4, version 1.7.0. Modulen är en öppen källkods -baserad modul, som installerades till Joomla. Weather GK4 hämtar väderinformationen med hjälp av en så kallad WOEID (Where On Earth IDentifier), från webbsidan GeoPlanet Explorer, http://isithackday.com/geoplanet-explorer/. Genom att skriva in namnet på staden, vars väderinformation man vill hämta, i sökfältet på webbsidan, hämtas informationen med hjälp av Yahoo Weather API (Application Programming Interface, applikations-programmeringsgränssnitt).

6.2.5 JCK Editor

JCK Editorn är ett WYSIWYG-verktyg utvecklat av WebXSolution Ltd och baserar sig på öppen källkod. WYSIWYG står för ”What You See Is What You Get”, alltså ”Vad Du Ser Är Vad Du Får”. Idén bakom användningen av ett sådant här verktyg är att lätt kunna skapa innehåll till webbsidan utan någon eller liten kunskap om HTML. Man kan även lätt administrera det här tillägget och på så sätt justera texteditorns egenskaper baserad på användarkategorier. Man har även möjlighet att skapa nya editorprofiler. Dock måste man skilt optimera verktyget för att det som skrivs skall stämma överens med det som visas åt allmänheten på webbsidan.

6.2.6 Google Maps

Google Maps är också ett verktyg baserat på öppen källkod. Den här funktionen användes på webbsidans kontaktsida för att peka ut var på kartan Västankvarn Gård ligger. Webbsidans besökare kan på så sätt lätt få kördirektiv, från t.ex. sin egen bostad, till gården utan större ansträngning. Google Maps implementerades till webbsidan med hjälp av en ”iframe” hämtad från Google Maps egen webbplats.

6.2.7 Webbshop

Webbshopen på webbsidan är en färdig installerbar komponent kallad HikaShop, som är skapad av Hikari. HikaShop är baserad på öppen källkod och är den bästa kundservicekomponenten, som skapats för Joomla CMS:et. Med komponenten kan man lätt lägga till och ta bort produkter från försäljning. Med komponenten följer även en kundvagnsmodul. Den här modulen är placerad på högra sidan av webbshopen på

webbsidan och uppdateras automatiskt när man lägger till en produkt till eller tar bort en produkt från kundvagnen.

6.2.8 Händelsekalender

Webbsidan har också en händelsekalender, som är en färdig modul kallad ”JEvents Calender”, utvecklad av GWE Systems Ltd. Med kalenderns funktioner kan man lätt justera olika inställningar så att modulen visar dag, månad, vecka och/eller år. Man kan även välja mellan sex stycken olika utseenden. Kalenders globala utseende valdes för att sedan modifiera koden så att kalendern färgmässigt passade in i webbsidans tema.

6.3 Kodandet av responsiviteten

Till skillnad från en statisk webbsida kräver en responsiv webbsida mer tankearbete och mer krävande kodning. Då man lägger upp en statisk webbsida med endast en bredd behövs kod för endast den specifika bredden för den webbsidan. Om man istället gör en responsiv webbsida där sidan anpassas enligt bredden på webbläsarfönstret, behövs det kod för att webbsidans utseende skall fungera oberoende av bredden.

En responsiv webbsida skall även leva och en användare skall kunna besöka sidan utan större problem. Webbsidan skall därför ta hänsyn till de tekniska enheter besökaren kan tänka sig använda. Med hjälp av responsiv webbdesign försöker man eliminera skrollning i horisontell riktning, vilket uppkommer om besökaren använder en lägre skärmupplösning än vad webbsidan är lämpad för.

6.3.1 LESS och Media Queries

För att koda responsiviteten användes kodspråket LESS. Till kodningen av responsiviteten och optimeringen av den, för olika skrämupplösningar samt terminaler, användes CSS3-funktionen ”media queries”. Användningen av ”media queries” illustreras i kodexempel 25.

Kodexempel 25. Förklaring till hur media queries används

```
@media screen and (max-width: 320px) {
/*utseendekoden för webbsidan på en max bredd av 320 pixlar skrivs mellan dessa taggar*/
}
```

De olika bredderna, som bestämmer utseendet, sattes till max 320 pixlar, medan minimum-bredderna vilar på 321, 480, 600 och 960 pixlar. Med maximumbredd (max-width) menas att sidan är mindre än eller lika med x-antal bildpunkter. Det samma gäller om minimumbredden (min-width) är specificerad, men då är bredden större än eller lika med minimumet. Webbsidans utseende enligt olika ”media query”-bredder illustreras i figur 7.

Figur 8. Bild på hur responsiv webbdesign fungerar

Sidan fungerar i alla bredder och inga element har lämnats bort oberoende av upplösning. De element, som visas i högra kanten av webbsidans innehållscontainer på en upplösning på 600 pixlar eller högre, visas under containern på lägre upplösningar. Detta gör att användaren kan se webbsidans alla element med vilken teknisk enhet som helst.

6.3.2 Navigationen

Ett av de viktigaste elementen på en webbsida är navigationen och hur den skall fungera, samt vara strukturerad, för att webbsidan skall vara så användarvänlig som möjlig på alla typer av terminaler. Därför är det mycket viktigt att menyn alltid är synlig och snabbt tillgänglig. En väldesignad webbsida har en organiserad meny och sidan blir på så sätt användarvänlig.

Ju bättre navigationen är på en webbsida, desto lättare har besökaren att hitta önskad information. Webbsidan bör inte heller vara byggd kring första sidan utan man skall kunna komma in på vilken sidan som helst och sedan enkelt kunna hitta till önskad information. (Klust Creative)

För att få menyn att bete sig responsivt används samma menymodul tre gånger, men varje modul renderas på olika sätt beroende på fönstrets bredd. För att åstadkomma detta

skapades tre modulpositioner i templateDetails.xml-filen, ”navigation_mobile”, ”navigation_resp” och ”navigation”. Vilken modulposition, som visas när, bestäms i LESS-koden genom att lägga till ”display: none;” på de divar med den modulposition, som inte skall renderas ut vid ett specifikt utseende.

För mobil- och surfplatteversionerna av webbsidan användes menyutseendet ”dropselect”, vilket gav en dropdown-liknande meny. Detta utseende visas på bredderna 320 till 550 pixlar och därefter byter menyns utseende till basutseende. Mobil- och surfplatteversionernas meny renderas i modulpositionerna ”navigation_mobile” och ”navigation_resp”. Menyutseendet avsett för datorskärmar renderas i ”navigation”-positionen. I figur 8 illustreras menyns utseende vid olika skärmupplösningar.

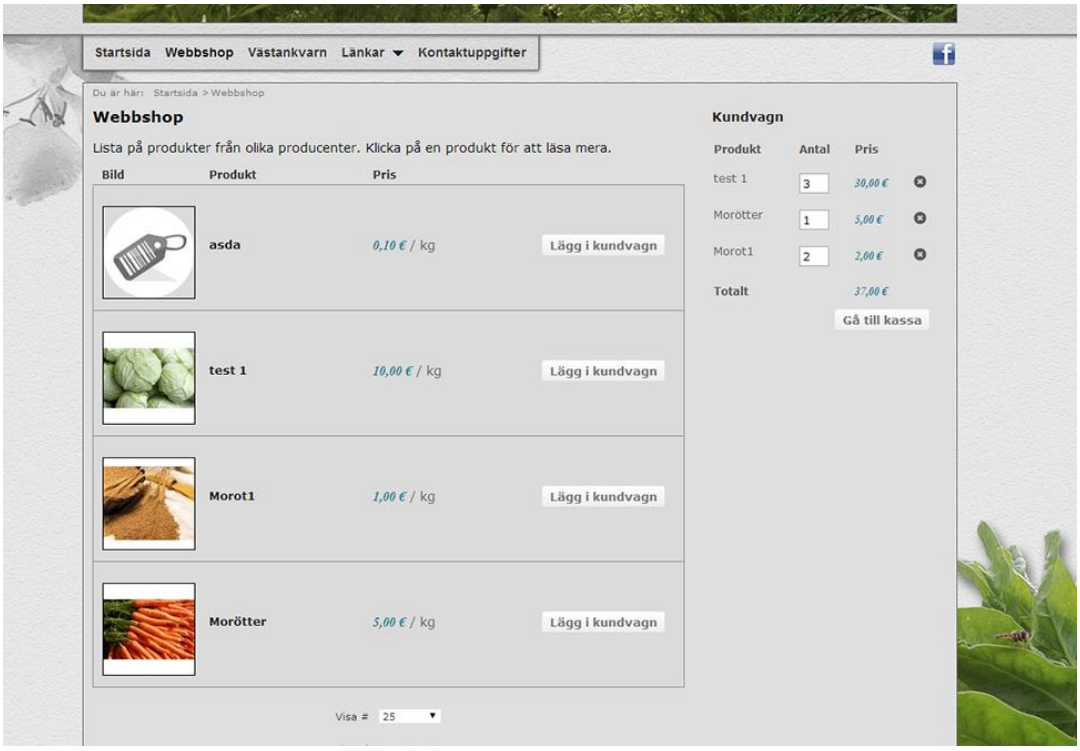
Figur 9. Menyns beteende vid olika upplösningar

Anledningen till att den här lösningen valdes istället för en responsiv meny, var att de färdiga modulerna för menyer med responsivitet kostade pengar. En sådan här typ av meny kan dock implementeras till webbsidan i ett senare skede, om man så önskar.

6.3.3 Webbshopen och kundvagnsmodulen

Från HikaShop-komponentens ”back-end” valdes tabellform för strukturen till webbshopens produktkatalog. Produkterna valdes att visas i tabellform, istället för div-format, eftersom de skulle vara uppradade i en lista under varandra. Det var även lättare att formatera datan i tabellen med LESS-kod, än det skulle ha varit i en div. I figur 9 illustreras områdena för webbshopen och kundvagnen.

Kundvagnsmodulen är placerad i den högra kolumnen och kommer därför att tryckas under webbshopen i lägre upplösningar. Detta på grund av att inte behöva göra allt så smått och ihoptryckt.



Figur 10. Webbshopens och kundvagnsmodulens utseenden (test version från lokal utveckling)

6.4 Webbfonter

Bland alla fonter, som används på Internet, ansågs Verdana vara den bästa. Verdana valdes, eftersom fonten är en ”sans-serif”-fonttyp. ”Sans-serif”-fonter används i webbdesign för att informationen på webbsidan även skall kunna läsas av dyslektiker.

6.5 Optimeringen för Internet Explorer

Optimeringen för Internet Explorer skedde endast med CSS-kod, eftersom LESS-komponenten endast tillåter att man använder en .less-fil. Webbsidans helhetsutseende optimerades även bara för versionerna 7 och 8 av Internet Explorer, detta eftersom dessa versioner inte stöder CSS3.

För att kontrollera om webbsidans användare besöker sidan med Internet Explorer eller någon annan webbläsare, skrevs en ”if IE”-kodsnutt in i temakoden, index.php. Den här koden går även att specificera så att den kontrollerar vilken version av Internet Explorer besökaren använder. En sådan här typ av kod, kallas ”conditional comments” och fungerar endast i Internet Explorer.

Koden kontrollerar först om besökaren använder sig av Internet Explorer version 7. Om påståendet i kodexempel 26 stämmer, skrivs innehållet i filen ie7.css ut och då ser webbsidan bra ut även i den här versionen.

Kodexempel 26. Exempel på ”if IE” -kontrollkod

```
<!—[if IE 7]>
    <link rel=”stylesheet” href=”css/ie7.css”>
<![endif]-- >
```

Optimeringsalternativen går ut på att först kontrollera om besökarens version av webbläsaren är äldre eller lika med 8. Här placerades koden, som fungerade lika i både version 7 och 8. Sedan kontrollerades det om versionen var 7 eller 8 och där sattes den CSS-kod som fungerade för respektive version av webbläsaren. Orsaken till att webbsidans utseende inte har optimerats skilt för Internet Explorer version 9 och senare, är att dessa versioner stöder CSS3, och på så sätt fungerar även responsiv webbdesign i dem.

Orsaken till att man måste optimera utseendet skilt för Internet Explorer är att den här webbläsaren läser och tolkar CSS-koden på ett annat sätt än Mozilla Firefox, Google Chrome och Opera. De mest besvärliga delarna för Internet Explorer är margin (yttre marginal), padding (inre marginal) och ramar.

6.6 Favikon

En favikon är en 16-pixlar hög och 16-pixlar bred bild, som visas i webbläsarens flik. Filformatet på bilden skall vara .ico för att bilden även skall fungera i äldre versioner av webbläsarna. För att skapa den här bilden användes Adobe Photoshop. Tyvärr kan man inte spara bilder som .ico i Photoshop, utan man måste konvertera filformatet med hjälp av ett online-konverteringsverktyg.

Webbplatsens favikon består av ett par morötter, tagna ur webbplatsens logo. Bilden har även genomskinlig bakgrund. Favikonen sparades i .png-format och filformatet konverterades sedan om med hjälp av onlinetjänsten http://favicon-generator.org/.

Placeringen av en favikon, i Joomla, skiljer sig en aning från andra system. I Joomla skall favikonen laddas upp till mappen ”/joomla/templates/<ditt tema>”, där ”ditt tema” är namnet på temat man använder på webbsidan. Medan man i normala fall laddar upp

favikonen till webbsidans rot-mapp på webbhotellet. Favikonen skall även döpas om till favicon.ico.

6.7 Användarmanualen

En av beställarens önskningar var att få en så tydlig och omfattande användarmanual som möjligt. Användarmanualen skrevs i slutet av projektet och språket var svenska. För att manualen inte skulle bli tråkig lades även beskrivande bilder till texten.

Manualen är skriven ur en persons, som inte har någon eller ganska liten teknisk erfarenhet, synvinkel och omfattar hur man använder basfunktionerna i webbsidans ”back-end”. Till basfunktionerna hör hur man skapar och tar bort artiklar och menylänkar, hur man editerar sådana och hur man använder sig av JCK Editorn. Även mer avancerade funktioner, såsom hur man lägger till och tar bort moduler och hur man redigerar dem, tas upp i manualen.

Den mest avancerade funktionen, som beskrivs i användarmanualen, är hur man använder webbshopens administrationsverktyg. Här beskrivs hur man lägger till och tar bort produkter och vad som händer med en produkt när den är slutsåld. I kapitlet om webbshopen kan man även läsa hur man redigerar ett produktpris och hur man flyttar produkten från en produktkategori till en annan.

7 Uppladdning av webbsidan

I slutskedet av arbetet laddades webbsidan upp till ett webbhotell för att kunna visas åt allmänheten. För uppladdning användes FTP-klienten FileZilla. Då Joomlas filer hade laddats upp till webbservern, installerades CMS:et och kontrollerades att det fungerade. Sedan installerades alla nödvändiga komponenter, moduler och insticksprogram för att åstadkomma samma funktioner, som hade fungerat i lokal utveckling. Joomla uppgraderades till version 3.2.3 från 3.2.2, som hade använts vid lokal utveckling. De senaste versionerna av moduler, komponenter och insticksprogram valdes också för att öka säkerheten på webbsidan. Efter detta lades det till artiklar, menylänkar och moduler aktiverades och justerades. Förekomsten av eventuella buggar eller grafiska problem kontrollerades också. Sist men inte minst skapades ett användarkonto åt beställaren så att hon skall kunna uppdatera webbsidan från ”back-enden”.

8 Problem som uppstått

Under utvecklingen uppstod några problem, men dessa kunde lösas med mindre förhinder och ansträngning. Redan i ett tidigt skede uppstod första problemet. Detta skedde efter att Joomla hade installerats lokalt på WAMP-serverprogrammet och installationen av de komponenter, som behövdes i det skedet av utvecklingen, hade skett. Problemet var en massa felmeddelanden vid översättning av en text med komponenten Falang. Först såg det ut som om Falang var problemet, men sedan visade det sig att Joomla inte var kapabelt att konstruera nya filer i databasen på grund av de dåvarande installationsinställningarna.

Problemet löstes med att, vid lokalinstallation, ändra inställningarna för databashanterarens motor i joomla.sql-filen i installationsmappen. Efter en ominstallation fungerade allt felfritt igen. Ändringen gällde databasmotorn och den var nödvändig eftersom Joomla nyligen övergick till att använda sig av InnoDB, som basinställning, istället för MyISAM. Efter att alla ”ENGINE=InnoDB” hade ändrats till ”ENGINE=MyISAM” kunde Joomla igen kontakta MySQL databasen och då också konstruera nya filer i den. De största skillnaderna mellan dessa är prestandan, samt säkrare integritet och transaktioner i InnoDB-systemet.

Att få bilderna på webbsidan att förminskas och förstoras i samband med förhållandet på webbläsarfönstret har också varit ett problem. För att lösa detta användes enkel LESS-kod där man definierar bildens bredd i procent. Då man definierar i procent kommer bildens bredd att påverkas i moderelementet, dvs. det element, som bilden är placerad i. Om bilden istället hade justerats enligt texten, skulle bildens bredd definierats i em i stället för procent.

Användning av CSS-kodens ”override”-funktion har också förekommit ett antal gånger. ”Override”-funktionen känns igen av ”!important” i slutet av en CSS-rad. !Important används då man vill köra över redan skriven kod och använda egen kod istället.

Vid utvecklingen av den responsiva webbdesignen uppstod ett större problem, som dock var lätt att åtgärda. Under optimeringen av webbsidans utseende för de mindre skärmupplösningarna uppstod det ett stort område till höger om webbsidans ”wrapper”-div. Detta berodde på att elementets bredd hade definierats i pixlar i stället för procent. Problemet åtgärdades genom att ändra bredden till procent i stället.

Ett annat problem, som inte hade tagits i beaktande från början, var att man skall kunna lägga till nya huvudlänkar till menyn. Detta blev ett problem i menypositionen, på grund av hur menyns utseendestruktur var upplagd. Strukturen var upplagd så att menyns bakgrundsbild, 40-pixlar hög och 500-pixlar bred, var placerad i en div kallad ”navbg” och själva menyn kom ovanpå den här diven. Detta resulterade i att den fixerade bakgrunden inte utvidgades eller minimerades horisontellt då man lade till eller tog bort en menylänk.

Intressantare problem har till exempel varit hur man får en fotnot att alltid vara klistrad till fönstrets nedre kant, och hur man arrangerar element så att de är framför eller bakom varandra.

8.1 Fotnoten

Anledningen till att fotnoten tas upp och berättas om skilt är att den orsakade problem i större skärmupplösningar. Det var meningen att webbsidans fotnot skulle vara klistrad, och såg ut att vara det vid utveckling av sidan i en mindre skärmupplösning än vad som normalt användes. Så snart webbsidan testades på en större skräm visade det sig att webbsidans bakgrund lyste igenom mellan fotnoten och webbläsarens nedre kant. Problemet löstes dock efter många försök och justeringar i LESS-koden.

9 Reflektioner

I följande punkter diskuteras lärdomar under utvecklingen av produkten för examensarbetet och vad som skulle ha gjorts annorlunda. Reflektioner över varför ett programmeringsspråk valdes över ett annat, tas också upp.

9.1 Lärdomar

Under tiden som webbsidan för projektet ”Västankvarn - En Västnyländsk matkälla” har utvecklats har problem uppstått och lösts genom egen erfarenhet eller Internet och andra människors kunnande, tips och råd. Lärdomar från den här tiden har bl.a. varit hur man ordnar element på ett effektivare sätt med hjälp av z-index, hur man använder sig av position-egenskapsvärden och vad det är för skillnad mellan dem. Även vad som behövs för att man skall kunna ge ett z-index-värde åt ett element och hur man använder dessa

värden i tidigare versioner av webbläsaren Internet Explorer har utökat kunskaperna i kodning.

Andra lärdomar som uppkommit är strukturkodning för att lättare se vilka element i LESS-koden som hänger ihop. Ju mer strukturerad man är desto lättare har man att hitta en specifik kod i framtiden. Det mest lärorika har ändå varit hur man utvecklar en webbsida med responsiv webbdesign. Tankesättet, som behövs för en sådan här typ av hemsida, är mycket mer krävande än då man utvecklar en statisk webbsida. Vid utveckling av responsiv webbdesign bör man utvidga sitt tänkande utanför ramarna, ”thinking outside the box”, som det heter på engelska.

En bra idé och struktur är viktiga hörnstenar för att ett projekt skall lyckas. De hjälper utvecklingen och kodningen av en webbsida. Idén skall också vara klar före man börjar överföra bilden, från en pappersskiss, till ett bildredigeringsprogram.

Som redan nämndes tidigare har ett problem varit hur man får en bild att justeras i förhållande till fönstret. I och med att information och lösningar då söktes lästes också en artikel om hur man använder CSS-kod till att skära bilder. så att endast en del av bilden visas beroende på vilken upplösning användare, som besöker sidan, använder sig av. Även om den här tekniken inte användes i det här projektet har det lagts på minnet och kommer förhoppningsvis att användas i framtida projekt.

Historien bakom de här programmeringsspråken och innehållshanteringssystemen har också varit intressant att lära sig. Orsakerna till varför man har gått över från ett sätt att koda till ett annat har också varit fascinerande. Likaså har för- och nackdelarna för utveckling av webbsidor och deras syften varit.

9.2 Varför valdes dessa programmeringsspråk och program?

Valet av programmeringsspråk och innehållshanteringssystem baserade sig på viljan att lära sig ett nytt sätt att använda Joomla. Användningen av kod för att underlätta arbetet intresserade också. En annan orsak till varför LESS valdes är att framtida justeringar och ändringar till koden underlättas.

Orsaken till att Joomla valdes är för att programmet är mycket användarvänligt och det finns en massa gratis moduler och komponenter som man kan använda. Systemets administrationspanel är också mycket välstrukturerad och det är lätt för en nybörjare att

skapa nytt innehåll, nya texter och menylänkar. Det är även lätt att lägga till och ta bort, samt justera block så att de endast visas på de sidor man vill. Vidare är det också lätt att modifiera olika moduler, med hjälp av PHP-kod, så att modulerna fungerar på önskat vis. Det är också lätt att använda egna språkinställningar istället för färdiga genom att ändra på hur ett textelement visas utan att behöva redigera en .ini-fil.

Inom webbutvecklingsbranschen krävs det hela tiden att man lär sig något nytt. Det nya sättet att koda och att de nya kodningsspråken är ofta uppräknade, som krav då man söker arbete. Från det att jag hörde om LESS för första gången hade jag tänkt använda det, men aldrig vågat eller haft tid att lära mig något nytt. Ett sådant här projekt och arbete gav mig ett ypperligt tillfälle att sätta mig in i ett nytt kodspråk och lära mig grunderna i det. LESS är mycket omfattande och det kommer att ta tid att lära sig använda det till dess fulla funktionalitet, men som ordspråket lyder ”övning ger färdighet”.

9.3 Vad skulle ha kunnat göras mer annorlunda?

Med tanke på arbetets omfattning borde en tilltalande design ha gjorts från första början. Dessutom borde arbetet också ha påbörjats tidigare. En genast tilltalande design skulle ha sparat en massa tid i slutskedet av projektet. Tyvärr är det så i webbutvecklingsbranschen att kunden inte alltid vet vad han eller hon vill ha före designern ger förslag eller redan gjort ett förslag till utseende. Detta borde ha tagits i beaktande redan i början och önskemål om funktioner samt färger till webbsidan borde ha lagts fram av båda parterna.

När det gäller utseendet borde en mörk och en ljus version av förslaget lagts fram så att kunden skulle ha haft möjlighet att välja mellan två olika versioner, och inte bara ha haft en version och sedan få klartecken för den versionen, bara för att ändra på den i ett senare skede. Detta är både tidskrävande och extremt opraktiskt för webbsidans utvecklare, eftersom tiden som krävs för att designa om ett utseende kan användas till annat.

Framställning av de olika versionerna av webbsidans utseende borde också ha skett i Adobe Photoshop, med tanke på arbetets svårighetsgrad. Genom att ha alla versioner grafiskt framställda underlättar kodningen. Den grafiska framställningen av endast webbsidans helhetsutseende resulterade i att kodningen försvårades märkbart. Fastän de lägre skärmupplösningarna framställdes endast via LESS, framskred arbetet riktigt bra. Själva slutversionen av utvecklingsprojektet blev också riktigt intressant och bättre än förväntningarna.

10 Sammanfattning

Då webbsidan var klar och allting fungerade, som det skulle, ordnades det ett möte med kunden. Under mötet visades webbsidan åt beställaren och hur dess responsiva webbdesign fungerade och feedbacken var positiv. Detta var första gången jag har lyckats skapa en responsiv webbdesign och det har faktiskt varit en lärorik upplevelse. I framtiden kommer den erfarenhet och de lärdomar jag har fått under detta projekt verkligen att vara till nytta.

Det viktigaste med att utveckla en webbsida är att både kunden och designern är nöjda med slutresultatet. Utvecklingen av webbplatsen har i stort sett hela tiden gått framåt. Dock har det förekommit några få problem under vägen som måste lösas. Den största motgången, under projektet, har varit omdesignandet av webbsidans utseende. Detta ledde till kodjustering för att få webbsidans nya utseende från bild till verklighet. Så här efteråt tänkt skulle den lättare lösningen kanske ha varit att koda om utseendet, från början, i stället för att justera den existerande koden. I de flesta fall sparar man tid om man kodar om i stället för att ändra på existerande kod.

Kundkontakten under projektet har skett antingen som direkt kontakt, kundmöten, eller via e-post. De gånger vi har haft ett kundmöte har feedback och korrigeringsönskningar getts angående webbsidan och vi har diskuterat arbetets gång. Detta har varit till stor hjälp för att hålla arbetet i rätt riktning och inte avvika från målet. Även motivationen att få arbetet färdigt har upprätthållits på det här sättet, men ibland saknades den vilket ledde till att det blev bråttom att färdigställa arbetet, eftersom tiden började ta slut.

Jag är nöjd med valet av CMS, inte bara för att jag redan hade erfarenhet av det, utan också eftersom det visade sig vara ett ypperligt och smidigt verktyg för att uppnå olika funktioner. Skapandet av ett eget tema, samt installering av olika slags moduler, komponenter och instickspprogram gick även smidigt och på så sätt sparade det en massa tid. Den sparade tiden kunde användas till att finslipa samt justera olika element i utseendet. Även uppladdningen av webbsidan till webbhotellet gick smärtfritt och sidan var snabbt tillgänglig för allmänheten.

Nu när jag tänker tillbaka på tiden då jag började med detta arbete är det några saker som skulle ha kunnat göras bättre. Till dessa hör skapandet av en exakt tidtabell, för att mer exakt kunna följa med hur arbetet flöt fram. Även om tiden blev knapp i slutändan och motivationsbrist fanns, blev slutversionen av webbsidan bättre än förväntad och vi var

båda nöjda med slutresultatet. Till sist kontrollerades webbdesignens responsivitet på en Nokia N8. Slutresultatet kan ses på addressen: http://vastankvarn-envastnylandskmatkalla.net/

11 Kundens synvinkel

”Uppdragets slutprodukt skulle vara en hemsida med webbshop, som användaren själv enkelt och snabbt kan uppdatera och det arbetet är utfört. Arbetet med hemsidan har från beställarens sida gått fint, kommunikationen har fungerat fint båda vägarna.

Under arbetets gång har regelbundna på förhand bestämda möten hållits och tillsammans har vi diskuterat lösningar som fungerar. Arbetet påbörjades i juni 2013 och uppskattades preliminärt vara klart i november 2013, hemsidan meddelades vara klar 1.5.2014.

Rasmus har tagit initiativ till och förklarat olika IB-tekniska lösningar, som kunde lämpa sig för detta arbete. Rasmus har också tagit reda på och delgett alternativ till bland annat webbhotell och utformning av visuell design. Som uppdragsgivare är jag belåten med slutprodukten, arbetssättet och jag är särskilt belåten över den goda kommunikationen.” (Ulrika Grönvik)

Källförteckning

Ullman, L (2002). *Visuell snabbguide PHP*. Harlow. Essex, England: Peachpit Press

Wyke-Smith, C (2006). *Stila med CSS – En guide för designers*. Sundbyberg: Pagina Förlags AB

Drupal (u.å), About Drupal [Online]
https://drupal.org/about [Hämtat 26.4.2014]

FileZilla (u.å), Overview [Online]
https://filezilla-project.org/ [Hämtat 17.4.2014]

Infomesh (u.å), The Early History of HTML [Online]
http://infomesh.net/html/history/early [Hämtat 20.4.2014]

Jared Chelf (20.3.2014), Why a Responsive Web Design Will Help You Get More Visitors [Online]
http://webdesignledger.com/tips/why-a-responsive-web-design-will-help-you-get-more-visitors [Hämtat 25.3.2014]

Jeremy Hixon (9.11.2011), An Introduction To LESS and Comparision to Sass [Online]
www.smashingmagazine.com/2011/09/09/an-introduction-to-less-and-comparison-to-sass/ [Hämtat 18.5.2014]

Joomla (u.å), Technical Requirements [Online]
http://www.joomla.org/technical-requirements.html [Hämtat 26.4.2014]

Klust Creative (u.å), Importance of Navigation in Web Design [Online]
http://klustcreative.com/theblog/importance-of-navigation-in-web-design
[Hämtat 26.3.2014]

lesscss (u.å), Color Blending Functions [Online]
http://lesscss.org/functions/#color-blending [Hämtat 18.5.2014]

lesscss (u.å), Color Operation Functions [Online]
http://lesscss.org/functions/#color-operations [Hämtat 18.5.2014]

lesscss (u.å), History [Online]
http://lesscss.org/about [Hämtat 20.4.2014]

Marvi Ocampo (u.å), 10 Reasons Why You Should Learn Adobe Photoshop [Online]
http://naldzgraphics.net/design-2/learn-adobe-photoshop/ [Hämtat 15.3.2014]

phpmyadmin (u.å), History [Online]
http://www.phpmyadmin.net/home_page/about.php [Hämtat 17.4.2014]

Rob Tomlinson (u.å), The Advantages of using a CMS. [Online]
http://www.rob-tomlinson.com/the-advantages-of-using-a-cms [Hämtat 7.4.2014]

Stackoverflow (u.å), Fancy Media Queries with some LESS Magic [Online]
http://stackoverflow.com/questions/15837808/fancy-media-queries-with-some-less-magic
[Hämtat 19.5.2014]

W3Techs (u.å), Usage of content management systems for websites [Online]
http://w3techs.com/technologies/overview/content_management/all [Hämtat 19.5.2014]

Webopedia (u.å), WAMP [Online]
www.webopedia.com/TERM/W/WAMP.html [Hämtat 17.4.2014]

Wikipedia (u.å), Adobe Photoshop [Online]
https://en.wikipedia.org/wiki/Adobe_Photoshop [Hämtat 10.11.2013]

Wikipedia (u.å), LESS [Online]
http://en.wikipedia.org/wiki/LESS [Hämtat 20.4.2014]

Wikipedia (u.å), Notepad++ [Online]
http://en.wikipedia.org/wiki/Notepad++ [Hämtat 17.4.2014]

Wikipedia (u.å), Sass (stylesheet language) [Online]
http://en.wikipedia.org/wiki/Sass_(stylesheet_language) [Hämtat 18.5.2014]

What is Joomla? (video, 2012)
http://www.youtube.com/watch?v=Qjnc0H8utks [Hämtad 30.4.2014]

WordPress (u.å), About WordPress [Online]
https://wordpress.org/about/ [Hämtat 26.4.2014]

Figur 1, Model-View-Controller-modellen (bild)
http://i.stack.imgur.com/brjhk.png

Figur 4, Responsive design (bild)
http://www.paulolyslager.com/wp-content/uploads/2011/10/responsive_design.png

Figurförteckning

Kodexempelförteckning

Tabellförteckning